Protocol independence through disjoint encryption under Exclusive-OR

Sreekanth Malladi

Dakota State University
Madison, SD 57042, USA
Sreekanth.Malladi@dsu.edu

May 11, 2010

Abstract. Multi-protocol attacks due to protocol interaction has been a notorious problem for security. Gutman-Thayer proved that they can be prevented by ensuring that encrypted messages are distinguishable across protocols, under a free algebra [1]. In this paper, we prove that a similar suggestion prevents these attacks under commonly used operators such as Exclusive-OR, that induce equational theories, breaking the free algebra assumption.

1 Introduction

It is quite common for users to simultaneously run multiple cryptographic protocols on their machines. For instance, a user might connect to a web site using `https` that uses the `SSL` protocol and also connect to another remote server using the `SSH` protocol. It is also quite common for a single protocol to consist of multiple sub-protocols.

A protocol might be secure when running in isolation, but not necessarily when running parallely with other protocols. In fact, Kelsey et al. [2] showed that, for any given secure protocol, it is always possible to create another protocol to break the original protocol. In an interesting practical study, Cremers analyzed 30 published protocols and reported that 23 of them were vulnerable to multi-protocol attacks [3]. Thus, they are a genuine and serious threat to protocol security.

In an outstanding work, Guttman-Thayer proved that, if encrypted messages are tagged with distinct protocol identifiers, multi-protocol attacks can be prevented [1]. For instance, if the notation $[t]_k$ denotes message t encrypted with key k, then encryptions in the `SSL` protocol should resemble $[\texttt{SSL}, t_1]_{k_1}$ and those in the `SSH` protocol should resemble $[\texttt{SSH}, t_2]_{k_2}$. With such tagging in place, it will not be possible for an attacker to replay encryptions across protocols, since users would check and verify the tags upon receipt of messages.

However, Guttman-Thayer considered a basic protocol model where operators for constructing messages (such as encryption algorithms) do not induce equations between syntactically different messages, such as $[t]_k = [k]_t$. Most "real-world" protocols such as SSL violate this assumption, and use operators that do induce equational theories, such as Exclusive-OR (XOR). It is extremely important to revisit Guttman-Thayer result under these operators, since such operators have been demonstrated to cause new attacks on protocols that are not possible under a free algebra [4].

This is the problem we consider in this paper: We prove that a tagging scheme that is similar to Guttman-Thayer's prevents multi-protocol attacks under the XOR operator and the ACUN theory induced by it. Our proof strategy is general, and could be used for other equational theories such as ACU,Idempotence and ACU,Inverse. We give some intuitions for this in our conclusion.

Organization. In Section 2, we introduce our formal framework including the term algebra, protocol model, constraint satisfaction, security properties and our main protocol design requirements. In Section 3, we prove a lynchpin lemma that we use in Section 4 to achieve the main result. We conclude with a discussion of future and related works.

2 The Framework

In this section, we formalize our framework to model and analyze protocols.

2.1 Term Algebra

We will start off with the term algebra. We derive much of our concepts here from Tuengerthal's technical report [5] where he has provided an excellent and clear explanation of equational unification.

We denote the *term algebra* as $T(F, \mathit{Vars})$, where Vars is a set of variables, and F is a set of function symbols or operators, called a *signature*. The terms in $T(F, \mathit{Vars})$ are called F-Terms. Further,

- $\mathit{Vars} \subset T(F, \mathit{Vars})$;
- $(\forall f \in F)(\mathsf{arity}(f) > 0 \wedge t_1, \ldots, t_n \in T(F, \mathit{Vars}) \Rightarrow f(t_1, \ldots, t_n) \in T(F, \mathit{Vars}))$.

The set of nullary function symbols are called *constants*. We assume that every variable and constant have a "type" such as Agent, Nonce etc.

We define F as $\mathit{StdOps} \cup \{\mathtt{XOR}\} \cup \mathit{Constants}$, where,

$$\mathit{StdOps} = \{\mathit{sequence}, \mathit{penc}, \mathit{senc}, \mathit{pk}, \mathit{sh}\}.$$

penc and *senc* denote asymmetric and symmetric encryption operators respectively. *pk* and *sh* denote public-key and shared-key operators respectively. We assume that they will always be used with one and two arguments respectively, that are of the type Agent.

We use some syntactic sugar in using some of these operators:

$$\begin{aligned}
\mathit{sequence}(t_1, \ldots, t_n) &= [t_1, \ldots, t_n], \\
\mathit{penc}(t, k) &= [t]_k^{\rightarrow}, \\
\mathit{senc}(t, k) &= [t]_k^{\leftrightarrow}, \\
\mathtt{XOR}(t_1, \ldots, t_n) &= t_1 \oplus \ldots \oplus t_n.
\end{aligned}$$

We will omit the superscripts $\leftrightarrow$ and $\rightarrow$ for encryptions if the mode of encryption is contextually irrelevant.

We will write "$t \ \mathsf{in}\ [t_1, \ldots, t_n]$" if $t \in \{t_1, \ldots, t_n\}$. We will write $t_i \prec_t t_j$ if $t_i, t_j \ \mathsf{in}\ t$, $t = [t_1, \ldots, t_n]$ and $i < j$.

We define the subterm relation as follows: $t \sqsubset t'$ iff $t' = f(t_1, \ldots, t_n)$ where $f \in F$ and $t \sqsubset t''$ for some $t'' \in \{t_1, \ldots, t_n\}$.

We will use functions $\mathit{Vars}()$, $\mathit{Constants}()$, and $\mathit{SubTerms}()$ on a single term or sets of terms, that return the variables, constants and subterms in them respectively. For instance, if T is a set of terms,

$$\mathit{SubTerms}(T) = \{t \mid (\exists t' \in T)(t \sqsubset t')\}.$$

We will now introduce equational theories and equational unification.

Definition 1. [Identity and Equational Theory] *Given a signature F, and set of variables Vars, a set of* identities E *is a subset of* $T(F, \mathit{Vars}) \times T(F, \mathit{Vars})$. *We denote an identity as $t \cong t'$ where t and t' belong to $T(F, \mathit{Vars})$. An* equational theory *(or simply a theory) $=_E$ is the least congruence relation on $T(F, \mathit{Vars})$, that is closed under substitution and contains E. i.e.,*

$$=_E := \left\{ R \mid \begin{array}{c} R \text{ is a congruence relation on } T(F, \mathit{Vars}), E \subseteq R, \text{and} \\ (\forall \sigma)(t \cong t' \in R \Rightarrow t\sigma \cong t'\sigma \in R) \end{array} \right\}$$

For the signature of this paper, we define two theories, STD and ACUN.

The theory STD for *StdOps*-Terms is based on a set of identities between syntactically equal terms, except for the operator *sh*:

$$\begin{aligned}
\{[t_1, \ldots, t_n] &\cong [t_1, \ldots, t_n], \\
h(t) &\cong h(t), \\
\mathit{sig}_k(t) &\cong \mathit{sig}_k(t), \\
\mathit{pk}(t) &\cong \mathit{pk}(t), \\
[t]_k &\cong [t]_k, \\
\mathit{sh}(t_1, t_2) &\cong \mathit{sh}(t_2, t_1)\}.
\end{aligned}$$

The theory ACUN is based on identities solely with the $\mathtt{XOR}$ ($\oplus$) operator:

$$\{t_1 \oplus (t_2 \oplus t_3) \cong (t_1 \oplus t_2) \oplus t_3, t_1 \oplus t_2 \cong t_2 \oplus t_1, t \oplus 0 \cong t, t \oplus t \cong 0\}.$$

We will now describe equational unification.

Definition 2. [Unification Problem, Unifier, Unification Algorithm] *If F is a signature and E is a set of identities, then an E-*Unification Problem *over F is a finite set of equations*

$$\Gamma = \left\{ s_1 \overset{?}{=}_E t_1, \ldots, s_n \overset{?}{=}_E t_n \right\}$$

*between F-terms. A substitution σ is called an E-*Unifier *for Γ if $(\forall s \overset{?}{=}_E t \in \Gamma)(s\sigma =_E t\sigma)$. $U_E(\Gamma)$ is the set of all E-Unifiers of Γ. A E-Unification Problem is called E-*Unifiable *iff $U_E(\Gamma) \neq \{\}$.*

A complete *set of E-Unifiers of an E-Unification Problem Γ is a set C of idempotent E-Unifiers of Γ such that for each $\theta \in U_E(\Gamma)$ there exists $\sigma \in C$ with $\sigma \geq_E \theta$, where $\geq_E$ is a partial order on $U_E(\Gamma)$.*

An *E-Unification Algorithm* takes an E-Unification Problem Γ and returns a finite, complete set of E-Unifiers.

Hence forth, we will abbreviate "Unification Algorithm" to UA and "Unification Problem" to UP.

Two theories $=_{E_1}$ and $=_{E_2}$ are *disjoint* if the signatures used in the identities E_1 and E_2 have no common operators. UAs for two disjoint theories may be combined to output the complete set of unifiers for UPs made using operators from both the theories, using Baader & Schulz Combination Algorithm (BSCA) [6].

BSCA first takes as input a $(E_1 \cup E_2)$-UP, say Γ, and applies some transformations on them to derive $\Gamma_{5.1}$ and $\Gamma_{5.2}$ that are sets of E_1-UPs and E_2-UPs respectively. It then combines the unifiers for $\Gamma_{5.1}$ and $\Gamma_{5.2}$ obtained using E_1-UA and E_2-UA respectively, to return the unifier(s) for Γ (see Appendix A, Def. 7). Further, if Γ is $(E_1 \cup E_2)$-Unifiable, then there exist $\Gamma_{5.1}$ and $\Gamma_{5.2}$ that are E_1-Unifiable and E_2-Unifiable respectively.

We give a more formal and detailed explanation of BSCA in Appendix A using an example UP, for the interested reader.

2.2 Protocol Model

Our protocol model is based on the strand space framework [7].

Definition 3. [Node, Strand, Protocol] *A* node *is a tuple $\langle \pm,\ t \rangle$ denoted $\pm t$ where $t \in T(F,\ Vars)$. A* strand *is a sequence of nodes. A* protocol *is a set of strands.*

For instance, consider the $\mathsf{NSL}_{\oplus}$ protocol [8]:

Msg 1. $A \to B : [N_A, A]_{pk(B)}$
Msg 2. $B \to A : [N_A \oplus B, N_B]_{pk(A)}$
Msg 3. $A \to B : [N_B]_{pk(B)}$

Then, $\mathsf{NSL}_{\oplus} = \{role_A, role_B\}$, where,

$$\begin{aligned} role_A &= [+[A, N_A]_{pk(B)}, -[N_A \oplus B, N_B]_{pk(A)}, +[N_B]_{pk(B)}], \text{ and} \\ role_B &= [-[A, N_A]_{pk(B)}, +[N_A \oplus B, N_B]_{pk(A)}, -[N_B]_{pk(B)}]. \end{aligned}$$

A *semi-bundle* S for a protocol P is a set of strands formed by applying substitutions to some of the variables in the strands of P: If P is a protocol, then, $\mathsf{semi\text{-}bundle}(S, P) \Rightarrow (\forall s \in S)((\exists r \in P; \sigma)(s = r\sigma))$.

For instance, $S = \{s_{a1}, s_{a2}, s_{b1}, s_{b2}\}$ below is a semi-bundle for the $\mathsf{NSL}_\oplus$ protocol with two strands per role of the protocol:

$$\begin{aligned} s_{a1} &= [+[a1, n_{a1}]_{pk(B1)}, -[n_{a1} \oplus B1, N_{B1}]_{pk(A1)}, +[N_{B1}]_{pk(B1)}], \\ s_{a2} &= [+[a2, n_{a2}]_{pk(B2)}, -[n_{a2} \oplus B2, N_{B2}]_{pk(A2)}, +[N_{B2}]_{pk(B2)}], \\ s_{b1} &= [-[A_3, N_{A3}]_{pk(b1)}, +[N_{A3} \oplus b1, n_{b1}]_{pk(A3)}, -[n_{b1}]_{pk(b1)}], \\ s_{b2} &= [-[A_4, N_{A4}]_{pk(b2)}, +[N_{A4} \oplus b2, n_{b2}]_{pk(A4)}, -[n_{b2}]_{pk(b2)}]. \end{aligned}$$

(*Note*: lower-case symbols are constants and upper-case are variables).

We will assume that every protocol has a set of variables that are considered "fresh variables" (e.g. Nonces and Session-keys). If P is a protocol, then, $\mathit{FreshVars}(P)$ denotes the set of fresh variables in P. We will call the constants substituted to fresh variables of a protocol in its semi-bundles as "fresh constants" and denote them as $\mathit{FreshCons}(S)$. i.e., If $\mathsf{semi\text{-}bundle}(S, P)$, then,

$$\mathit{FreshCons}(S) = \left\{ x \mid \begin{pmatrix} \exists r \in P; s \in S; \\ \sigma; X \end{pmatrix} \begin{pmatrix} (r\sigma = s) \wedge (X \in \mathit{FreshVars}(P)) \wedge \\ (x = X\sigma) \wedge (x \in \mathit{Constants}) \end{pmatrix} \right\}.$$

We assume that some fresh variables are "secret variables" and denote them as $\mathit{SecVars}(P)$. We define "$\mathit{SecCons}()$" to return "secret constants" that were used to instantiate secret variables of a protocol: If $\mathsf{semi\text{-}bundle}(S, P)$, then,

$$\mathit{SecCons}(S) = \left\{ x \mid \begin{pmatrix} \exists r \in P; s \in S; \\ \sigma; X \end{pmatrix} \begin{pmatrix} (r\sigma = s) \wedge (X \in \mathit{SecVars}(P)) \wedge \\ (x = X\sigma) \wedge (x \in \mathit{Constants}) \end{pmatrix} \right\}.$$

For instance, N_A and N_B are secret variables in the $\mathsf{NSL}_\oplus$ protocol and $n_{a1}, n_{a2}, n_{b1}, n_{b2}$ are the secret constants for its semi-bundle above.

We will lift the functions $\mathit{Vars}()$, $\mathit{Constants}()$, and $\mathit{SubTerms}()$ to strands, protocols and semi-bundles. For instance, if P is a set of strands and $r \in P$, then,

$$\begin{aligned} &\mathit{SubTerms}(r) = \{t \mid (\exists t')((\langle \_, \ t' \rangle \text{ in } r) \wedge (t \in \mathit{SubTerms}(t')))\}, \\ &\mathit{SubTerms}(P) = \{t \mid (\exists r \in P)(t \in \mathit{SubTerms}(r))\}. \end{aligned}$$

We also define the long-term shared-keys of P as $\mathit{LTKeys}(P)$, where,

$$\mathit{LTKeys}(P) = \{x \mid (\exists A, B)((x = sh(A, B)) \wedge (x \in \mathit{SubTerms}(P)))\}.$$

To achieve our main result, we need to make some assumptions. Most of our assumptions are reasonable, not too restrictive for protocol design and in fact, good design practices.

As noted in [9], we first need an assumption that long-term shared-keys are never sent as part of the messages in the protocol, but only used as encryption keys. Obviously, this is a safe and prudent design principle.

Without this assumption, there could be multi-protocol attacks even when Guttman-Thayer suggestion of tagging encryptions is followed. For instance, consider the following protocols:

$\mathbf{P_1}$	$\mathbf{P_2}$
1. $a \rightarrow s : sh(a, s)$	1. $a \rightarrow b : [1, n_a]_{sh(a,s)}$

Now the message in the second protocol could be decrypted and n_a could be derived when it is run with the first protocol.

To formalize this assumption, we define a relation *interm* denoted $\Subset$ on terms such that,

- $t \Subset t'$ if $t = t'$,
- $t \Subset [t_1, \ldots, t_n]$ if $(t \Subset t_1 \vee \ldots \vee t \Subset t_n)$,
- $t \Subset [t']_k$ if $(t \Subset t')$,
- $t \Subset t_1 \oplus \ldots \oplus t_n$ if $(t \Subset t_1) \vee \ldots \vee (t \Subset t_n)$.

Notice that an interm is also a subterm, but a subterm is not necessarily an interm. For instance, n_a is an interm and a subterm of $n_a \oplus [a]^{\rightarrow}_{n_b}$, while n_b is a subterm, but not an interm.

Interms are useful in referring to the plain text of encryptions or everything that can be "read" by the recipient of a term. Contrast these with the keys of encrypted terms, which can only be confirmed by decrypting with the corresponding inverses, but cannot be read (unless included in the plain-text).

Assumption 1 *If P is a protocol, then, there is no term of P with a long-term key as an interm:*

$$(\forall t \in SubTerms(P))((\nexists t' \Subset t)(t' \in LTKeys(P))).$$

It turns out that this assumption is not sufficient. As noted by an anonymous reviewer of this workshop, we also need another assumption that if a variable is used as a subterm of a key, then there should be no message in which that variable is sent in plain (since a long-term shared-key could be substituted to the variable as a way around the previous assumption).

Hence, we state our next assumption as follows:

Assumption 2 *If $[t]_k$ is a subterm of a protocol, then no interm of k is an interm of the protocol:*

$$(\forall [t]_k \in SubTerms(P))((\nexists X \Subset k; t' \in SubTerms(P))(X \Subset t')).$$

Next, we will make some assumptions on the initial intruder knowledge. We will denote the set of terms known to the intruder before protocols are run, *IIK*. We will first formalize the assumption that he knows the public-keys of all the agents:

Assumption 3 *$(\forall x \in \mathit{Constants})(pk(x) \in \mathit{IIK})$.*

In addition, we will also assume that the attacker knows the values of all the constants that were substituted by honest agents for all the non-fresh variables (e.g. agent identities a, b etc.), when they form semi-strands:

Assumption 4 *Let P be a protocol. Then,*

$$(\forall x/X \in \sigma; r \in P)\left(\left(\begin{array}{c}\mathsf{semi\text{-}bundle}(S,P) \wedge (r\sigma \in S) \wedge \\ (x \in \mathit{Constants}) \wedge (X \notin \mathit{FreshVars}(P))\end{array}\right) \Rightarrow (x \in \mathit{IIK})\right).$$

Finally, we make another conventional assumption about protocols, namely that honest agents do not reuse fresh values such as nonces and session-keys:

Assumption 5 *Let S_1, S_2 be two different semi-bundles. Then,*

$$\mathit{FreshCons}(S_1) \cap \mathit{FreshCons}(S_2) = \{\}.$$

2.3 Constraints and Satisfiability

In this section, we will formalize the concepts given in [10,11] to generate symbolic constraints from node interleavings of semi-bundles and the application of reduction rules to determine satisfiability of those constraints.

Definition 4. [Constraints, Constraint sequences] *A* constraint *is a tuple $\langle m,\ T\rangle$ denoted $m : T$, where m is a term called the* target *and T is a set of terms called the* term set*. If S is a semi-bundle, then, cs is a* constraint sequence *of S, or $\mathsf{conseq}(cs, S)$ if every target term in cs is from a $-$ node of S and every term in every term set in cs is from a $+$ node of S.*

A constraint sequence cs is *simple* or $\mathsf{simple}(cs)$ if all the targets are variables. Constraint c is an "active constraint" of a constraint sequence cs (denoted $\mathsf{act}(c, cs)$) if all its prior constraints in cs, but not itself, are simple constraints. We denote the sequences before and after the active constraint of a sequence cs as $cs_<$ and $cs_>$ respectively.

In Table 1, we define a set of symbolic reduction rules, *Rules*, that can be applied on the active constraint of a constraint sequence.

concat	$[t_1, \ldots, t_n] : T$	$t_1 : T, \ldots, t_n : T$	split	$t : T \cup [t_1, \ldots, t_n]$	$t : T \cup t_1 \cup \ldots \cup t_n$
penc	$[m]_k^{\rightarrow} : T$	$k : T, m : T$	pdec	$m : [t]_{pk(\epsilon)}^{\rightarrow} \cup T$	$m : t \cup T$
senc	$[m]_k^{\leftrightarrow} : T$	$k : T, m : T$	sdec	$m : [t]_k^{\leftrightarrow} \cup T$	$k : T, m : T \cup \{t, k\}$
$\mathsf{xor_r}$	$m : T \cup t_1 \oplus \ldots \oplus t_n$	$t_2 \oplus \ldots \oplus t_n : T,$ $m : T \cup t_1$	$\mathsf{xor_l}$	$t_1 \oplus \ldots \oplus t_n : T$	$t_2 \oplus \ldots \oplus t_n : T,$ $t_1 : T$

Table 1. Set of reduction rules, *Rules*

The first column is the name of the rule, the second and third columns are the active constraints before and after the application of the rule.

We define a predicate $\mathsf{appl}()$ on each of these rules, that is true if the rule under consideration is applicable on the active constraint of the given constraint sequence. The predicate takes the name of the rule, the input sequence cs, the output sequence cs', input substitution σ, output substitution σ', and the theory Th considered as arguments. For instance, we define $\mathsf{xor_r}$ as follows[1]:

$$\mathsf{appl}(\mathsf{xor_r}, cs, cs', \sigma, \sigma', Th) \Leftrightarrow (\exists m, T, t) \begin{pmatrix} \mathsf{act}(m : T \cup t_1 \oplus \ldots \oplus t_n, cs) \wedge (\sigma' = \sigma) \wedge \\ (cs' = cs_<^\frown [t_2 \oplus \ldots \oplus t_n : T, m : T \cup t_1]^\frown cs_>) \end{pmatrix}$$

We left out two important rules in the table, un and ksub, that change the attacker substitution through unification. We describe them next:

$$\mathsf{appl}(\mathsf{un}, cs, cs', \sigma, \sigma', Th) \Leftrightarrow (\exists m, T, t) \begin{pmatrix} \mathsf{act}(m : T \cup t, cs) \wedge (cs' = cs_< \tau^\frown cs_> \tau) \wedge \\ (\sigma' = \sigma \cup \tau) \wedge (\tau \in U_E(\{m \overset{?}{=}_E t\})) \end{pmatrix}$$

$$\mathsf{appl}(\mathsf{ksub}, cs, cs', \sigma, \sigma', Th) \Leftrightarrow (\exists m, T, t) \begin{pmatrix} \mathsf{act}(m : T \cup [t]_k^\rightarrow, cs) \wedge \\ (cs' = cs_< \tau^\frown [m\tau : T\tau \cup [t]_k^\rightarrow \tau]^\frown cs_> \tau) \wedge \\ (\sigma' = \sigma \cup \tau) \wedge (\tau \in U_E(\{k \overset{?}{=}_E pk(\epsilon)\})) \end{pmatrix}$$

(*Note*: ϵ is a constant of type *Agent* representing the name of the attacker).

We will say that a constraint sequence cs' is a *child constraint sequence* of another sequence cs, if it can be obtained after applying some reduction rules on cs:

$$\mathsf{childseq}(cs, cs', Th) \Leftrightarrow (\exists r_1, \ldots, r_n \in Rules) \begin{pmatrix} \mathsf{appl}(r_1, cs, cs_1, \sigma, \sigma_1, Th) \wedge \\ \mathsf{appl}(r_2, cs_1, cs_2, \sigma_1, \sigma_2, Th) \wedge \ldots \wedge \\ \mathsf{appl}(r_n, cs_{n-1}, cs', \sigma_{n-1}, \sigma_n, Th) \end{pmatrix}.$$

We now define "normal" constraint sequences, where the active constraint does not have sequences on the target or in the term set and has stand-alone variables in the term set (also recall that by definition, the target term of an active constraint is not a variable):

$$\mathsf{normal}(cs) \Leftrightarrow \begin{pmatrix} \mathsf{act}(m : T, cs) \wedge \\ (\nexists t_1, \ldots, t_n)([t_1, \ldots, t_n] = m) \wedge \\ ((\forall t \in T)((\nexists t_1, \ldots, t_n)([t_1, \ldots, t_n] = t)) \wedge \\ (\forall t \in T)(t \notin Vars)) \end{pmatrix}$$

Next, we will define a recursive function, *normalize*(), that maps constraints to constraint sequences such that:

$$\begin{aligned} normalize(m : T) &= [m : T], \text{ if } \mathsf{normal}(m : T); \\ &= normalize(t_1 : T)^\frown \ldots ^\frown normalize(t_n : T) \text{ if } m = [t_1, \ldots, t_n]; \\ &= normalize(m : T' \cup t_1 \cup \ldots \cup t_n) \text{ if } T = T' \cup [t_1, \ldots, t_n]. \end{aligned}$$

[1] $^\frown$ is the sequence concatenation operator.

We will now overload this function to apply it on constraint sequences as well:

$$\begin{aligned} normalize(cs) &= cs, \text{ if } \mathsf{normal}(cs) \\ &= cs_{<}^{\frown} normalize(c)^{\frown} cs_{>}, \text{ if } \mathsf{act}(c, cs). \end{aligned}$$

We define satisfiability of constraints as a predicate "$\mathsf{satisfiable}$" which is true if there is a sequence of applicable rules which reduce a given normal constraint sequence cs to a simple constraint sequence cs_n, in a theory Th, resulting in a substitution σ_n:

$$\begin{array}{l} \mathsf{satisfiable}(cs, \sigma_n, Th) \Rightarrow \\ (\exists r_1, \ldots, r_n \in Rules) \left(\begin{array}{l} \mathsf{appl}(r_1, cs, cs_1, \{\}, \sigma_1, Th) \wedge \\ \mathsf{appl}(r_2, cs'_1, cs_2, \sigma_1, \sigma_2, Th) \wedge \ldots \wedge \\ \mathsf{appl}(r_n, cs'_{n-1}, cs_n, \sigma_{n-1}, \sigma_n, Th) \wedge \\ \mathsf{simple}(cs_n) \wedge \\ (\forall i \in \{1, \ldots, n\})(cs'_i = normalize(cs_i)) \end{array} \right). \end{array} \tag{1}$$

Notice the last clause which requires that every constraint sequence be normalized before any rule is applied, when checking for satisfiability.

This definition of satisfiability may seem unusual, especially for the puritans, since satisfiability is usually defined using attacker capabilities as operators on sets of ground terms to generate each target on constraints.

However, it was proven in [11] that the decision procedure on which our definition is based, is sound and complete with respect to attacker capabilities on ground terms in the presence of the algebraic properties of `XOR`. Hence, we defined it directly in terms of the decision procedure, since that is what we will be using to prove our main theorem. We refer the interested reader to [10] and [11] for more details on the underlying attacker operators, whose usage is equated to the decision procedure that we have used.

Note also that our definition only captures completeness of the decision procedure wrt satisfiability, not soundness, since that is the only aspect we need for our proofs in this paper.

2.4 Security properties and attacks

Every security protocol is designed to achieve certain security goals such as key establishment and authentication. Correspondingly, every execution of a protocol is expected to satisfy some related security properties. For instance, a key establishment protocol should not leak the key being established, which would be a violation of secrecy. It should also not lead an honest agent to exchange a key with an attacker, which would be a violation of both secrecy and authentication.

Our main result is general and is valid for any trace property such as secrecy, that can be tested by embedding the desired property into semi-bundles and then checking if constraint sequences from the semi-bundles are satisfiable:

Definition 5. [Secrecy]

A protocol is secure for secrecy *in the theory Th, if no constraint sequence from any semi-bundle of the protocol is satisfiable, after a strand with node that receives a secret constant is added to the semi-bundle. i.e., if P is a protocol, then,*

$$(\nexists sec, \mathsf{cs}, S)\begin{pmatrix}\mathsf{semi\text{-}bundle}(S, P) \wedge \mathsf{conseq}(\mathsf{cs}, S)\wedge \\ (\mathsf{cs} = [\_ : \_, \ldots, \_ : T])\wedge \\ (sec \in \mathit{SecCons}(S))\wedge \\ \mathsf{satisfiable}(\mathsf{cs}^\frown[sec : T], \sigma, \mathit{Th})\end{pmatrix} \Leftrightarrow \mathsf{secureForSecrecy}(P, \mathit{Th}).$$

2.5 Main Requirement - μ-NUT

We now formulate our main requirement on protocol messages to prevent multi-protocol attacks, namely μ-NUT, in the $\mathsf{S}\cup\mathsf{A}$ theory (an abbreviation for $\mathsf{STD}\cup\mathsf{ACUN}$). The requirement is an extension of Guttman-Thayer's suggestion to make encrypted terms distinguishable across protocols, to include XOR as well.

We will first define a set *XorTerms* as:

$$\{t \mid (\exists t_1, \ldots, t_n \in T(F, \mathit{Vars}))(t_1 \oplus \ldots \oplus t_n = t)\}.$$

We will also define a function *EncSubt*() that returns all the encrypted subterms of a set of terms. i.e., If T is a set of terms, then, $\mathit{EncSubt}(T)$ is the set of all terms such that if t belongs to the set, then t must be a subterm of T and is an encryption:

$$\mathit{EncSubt}(T) = \{t \mid (\exists t', k')((t = [t']_{k'}) \wedge (t \in \mathit{SubTerms}(T)))\}.$$

Further, if P is a protocol, then

$$\mathit{EncSubt}(P) = \{t \mid t \in \mathit{EncSubt}(\mathit{SubTerms}(P))\}.$$

We are now ready to state the main requirement formally:

Definition 6. [μ-NUT]

Two protocols P_1 and P_2 are μ-NUT-Satisfying, i.e., μ-NUT-Satisfying(P_1, P_2) iff:

1. *Encrypted subterms in both protocols are not STD-Unifiable after applying any substitutions to them:*

$$(\forall t_1 \in \mathit{EncSubt}(P_1), t_2 \in \mathit{EncSubt}(P_2))((\nexists \sigma_1, \sigma_2)(t_1\sigma_1 =_{\mathsf{STD}} t_2\sigma_2)).$$

2. *Subterms of XOR-terms of one protocol (that are not XOR-terms themselves), are not STD-Unifiable with any subterms of XOR-terms of the other protocol (that are not XOR-terms as well):*

$$\begin{pmatrix}\forall t_1 \oplus \ldots \oplus t_n \in \mathit{SubTerms}(P_1), \\ t'_1 \oplus \ldots \oplus t'_n \in \mathit{SubTerms}(P_2); t, t'\end{pmatrix}\begin{pmatrix}(t \in \{t_1, \ldots, t_n\}) \wedge (t' \in \{t'_1, \ldots, t'_n\}) \\ (t_1, \ldots, t_n, t'_1, \ldots, t'_n \notin \mathit{XorTerms})\wedge \\ \Rightarrow (\nexists \sigma, \sigma')(t\sigma =_{\mathsf{STD}} t'\sigma')\end{pmatrix}.$$

The first requirement is the same as Guttman-Thayer suggestion. The second requirement extends it to the case of `XOR`-terms, which is our stated extension in this paper.

The $\mathsf{NSL}_\oplus$ protocol can be transformed to suit this requirement by tagging its encrypted messages as follows:

Msg 1. $A \to B : [\mathsf{nsl}_\oplus, N_A, A]_{pk(B)}$
Msg 2. $B \to A : [\mathsf{nsl}_\oplus, [\mathsf{nsl}_\oplus, N_A] \oplus [\mathsf{nsl}_\oplus, B], N_B]_{pk(A)}$
Msg 3. $A \to B : [\mathsf{nsl}_\oplus, N_B]_{pk(B)}$

The constant "$\mathsf{nsl}_\oplus$" inside the encryptions can be encoded using some suitable bit-encoding when the protocol is implemented. Obviously, other protocols must have their encrypted subterms start with the names of those protocols.

3 A Lynchpin Lemma

In this section, we provide a useful lemma that is the lynchpin in achieving our main result. We prove in the lemma that, if we follow BSCA for $(\mathsf{S} \cup \mathsf{A})$-UPs that do not have `XOR` terms with variables, their ACUN subproblems will have only constants as subterms.

Lemma 1. [**ACUN UPs have only constants**]

Let $\Gamma = \{m \overset{?}{=}_{\mathsf{S}\cup\mathsf{A}} t\}$ be a $(\mathsf{S} \cup \mathsf{A})$-UP that is $(\mathsf{S} \cup \mathsf{A})$-Unifiable, and where no subterm of m or t is an `XOR` term with free variables[2]*:*

$$(\forall x)\left(\begin{array}{c}((x \sqsubset m) \vee (x \sqsubset t)) \wedge (n \in \mathbb{N}) \wedge \\ (x = x_1 \oplus \ldots \oplus x_n)\end{array} \Rightarrow (\forall i \in \{1, \ldots, n\})(x_i \notin \mathit{Vars})\right).$$

Then,

$$(\forall m' \overset{?}{=}_{\mathsf{ACUN}} t' \in \Gamma_{5.2}; y)\left(\left(\begin{array}{c}((y \sqsubset m') \vee (y \sqsubset t'))\wedge \\ (m' =_{\mathsf{ACUN}} t')\end{array}\right) \Rightarrow (y \in \mathit{Constants})\right).$$

Proof. Please see Appendix B, Lemma 2.

4 Main result - μ-NUT prevents multi-protocol attacks

We will now prove that μ-NUT-*Satisfying* protocols are not susceptible to multi-protocol attacks.

The idea is to show that if a protocol is secure in isolation, then it is in combination with other protocols with whom it is μ-NUT-*Satisfying*.

To show this, we will achieve a contradiction by attempting to prove the contrapositive. i.e., if there is a breach of secrecy for a protocol in combination with another protocol with which it is μ-NUT-*Satisfying*, then it must also have a breach of secrecy in isolation.

We assume that the reader is familiar with BSCA (detailed description in Appendix A).

[2] $\mathbb{N}$ is the set of natural numbers.

Theorem 1. *If a protocol is secure for secrecy, then it remains so in combination with any other protocol with which it is μ-NUT-Satisfying.*

Proof. Suppose P_1 is a protocol that is secure for secrecy in isolation in the $\mathsf{S} \cup \mathsf{A}$ theory. i.e., $\mathsf{secureForSecrecy}(P_1, \mathsf{S} \cup \mathsf{A})$. Consider another protocol P_2 such that, μ-NUT-$\mathit{Satisfying}(P_1, P_2)$. Let, S_1 and S_2 be two semi-bundles from P_1 and P_2 respectively:

$$\mathsf{semi\text{-}bundle}(S_1, P_1) \wedge \mathsf{semi\text{-}bundle}(S_2, P_2).$$

Consider a constraint sequence *combcs* from $S_{comb} = S_1 \cup S_2$. i.e.,

$$\mathsf{conseq}(\mathit{combcs}, S_{comb}).$$

Consider another constraint sequence *isocs*, where,
(a) Targets in *combcs* are targets in *isocs* if the targets belong to S_1:

$$(\forall m : \_ \ \mathsf{in}\ \mathit{combcs})((m \in \mathit{Terms}(S_1)) \Rightarrow (m : \_ \ \mathsf{in}\ \mathit{isocs})). \tag{2}$$

(b) Term sets in *combcs* are term sets in *isocs* but without terms from S_2:

$$\begin{pmatrix} \forall m_1 : T_1, \\ m_2 : T_2 \ \mathsf{in}\ \mathit{combcs} \end{pmatrix} \left(\begin{array}{c} m_1 : T_1 \prec_{combcs} m_2 : T_2 \\ \Rightarrow \\ (\exists T_1', T_2') \left(\begin{array}{c} (m_1 : T_1' \prec_{isocs} m_1 : T_2') \wedge \\ (T_1' = T_1 \setminus T_1'') \wedge (T_2' = T_2 \setminus T_2'') \\ (\forall t \in T_1'' \cup T_2'')(t \in \mathit{SubTerms}(S_2)) \end{array} \right) \end{array} \right). \tag{3}$$

Then, from Def. 4 (**Constraints**) we have: $\mathsf{conseq}(\mathit{isocs}, S_1)$.

Suppose *combcs* and *isocs* are normalized. To achieve a contradiction, let there be a violation of secrecy in S_{comb} s.t. *combcs* is satisfiable after an artificial constraint with a secret constant of S_1, say *sec*, is added to it:

$$(\mathit{combcs} = [\_ : \_, \ldots, \_ : T]) \wedge \mathsf{satisfiable}(\mathit{combcs}^\frown[\mathit{sec} : T], \_, \mathsf{S} \cup \mathsf{A}). \tag{4}$$

Suppose $[r_1, \ldots, r_n] = R$, such that $r_1, \ldots, r_n \in \mathit{Rules}$. Then, from the definition of satisfiability (1), using R, say we have:

$$\begin{pmatrix} (\mathit{combcs} = [\_ : \_, \ldots, \_ : T]) \wedge \\ \mathsf{appl}(r_1, \mathit{combcs}^\frown[\mathit{sec} : T], \mathit{combcs}_1, \{\}, \sigma_1, \mathsf{S} \cup \mathsf{A}) \wedge \\ \mathsf{appl}(r_2, \mathit{combcs}_1', \mathit{combcs}_2, \sigma_1, \sigma_2, \mathsf{S} \cup \mathsf{A}) \wedge \ldots \wedge \\ \mathsf{appl}(r_n, \mathit{combcs}_{n-1}', \mathit{combcs}_n, \sigma_{n-1}, \sigma_n, \mathsf{S} \cup \mathsf{A}) \wedge \\ \mathsf{simple}(\mathit{combcs}_n) \wedge (\forall i \in \{1, \ldots, n\})(\mathit{combcs}_i' = \mathit{normalize}(\mathit{combcs}_i)) \end{pmatrix}. \tag{5}$$

From their descriptions, every rule in *Rules* adds subterms of existing terms (if any) in the target or term set of the active constraint:

$$\begin{pmatrix} \mathsf{appl}(\_, cs, cs', \_, \_, \_) \wedge \mathsf{act}(m : T, cs) \wedge \\ \mathsf{act}(m' : T', cs') \wedge (x \in T' \cup \{m'\}) \end{pmatrix} \Rightarrow (x \in \mathit{SubTerms}(T \cup \{m\})). \quad (6)$$

Since every combcs'_i ($i = 1$ to n) in (5) is normalized, and since P_1 and P_2 are μ-NUT-*Satisfying*, we have:

$$(\forall i \in \{1, \ldots, n\}; \exists p \in \mathbb{N}; t_1, \ldots, t_p) \begin{pmatrix} \mathsf{act}(m : T, \mathit{combcs}'_i) \wedge \\ (t_1 \oplus \ldots \oplus t_p \in T \cup \{m\}) \Rightarrow \\ (\forall j \in \{1, \ldots, p\})(t_j \notin \mathit{Vars}) \end{pmatrix}. \quad (7)$$

Suppose *chcombcs* is a normal, child constraint sequence of *combcs* and *chisocs* is a normal, child constraint sequence of *isocs*.

Now all the rules in *Rules* are applicable on the target of the active constraint of *chisocs*, if they were on *chcombcs*, provided they are applied on a term of S_1:

$$\begin{array}{l} (\forall r \in \mathit{Rules}) \begin{pmatrix} \mathsf{appl}(r, \mathit{chcombcs}, \mathit{chcombcs}', \_, \_, \mathsf{S} \cup \mathsf{A}) \wedge \\ \mathsf{act}(m : \_, \mathit{chcombcs}) \wedge \mathsf{act}(m' : \_, \mathit{chcombcs}') \wedge \\ \mathsf{act}(m : \_, \mathit{chisocs}) \end{pmatrix} \Rightarrow \\ \left(\mathsf{appl}(r, \mathit{chisocs}, \mathit{chisocs}', \_, \_, \mathsf{S} \cup \mathsf{A}) \wedge \mathsf{act}(m' : \_, \mathit{chisocs}') \right). \end{array} \quad (8)$$

Similarly, all rules that are applicable on a term in the term set of the active constraint in *chcombcs*, say c, are also applicable on the same term of the active constraint in *chisocs*, say c' (provided the term exists in the term set of c', which it does from (3) and (6)):

$$\begin{array}{l} (\forall r \in \mathit{Rules}) \begin{pmatrix} \mathsf{appl}(r, \mathit{chcombcs}, \mathit{chcombcs}', \_, \_, \mathsf{S} \cup \mathsf{A}) \wedge \\ \mathsf{act}(\_ : \_ \cup t, \mathit{chcombcs}) \wedge \mathsf{act}(\_ : \_ \cup T', \mathit{chcombcs}') \wedge \\ \mathsf{act}(\_ : \_ \cup t, \mathit{chisocs}) \end{pmatrix} \Rightarrow \\ \left(\mathsf{appl}(r, \mathit{chisocs}, \mathit{chisocs}', \_, \_, \mathsf{S} \cup \mathsf{A}) \wedge \mathsf{act}(\_ : \_ \cup T', \mathit{chisocs}') \right). \end{array} \quad (9)$$

un and ksub are the only rules that affect the attacker substitution. We will show that these are equally applicable on *chcombcs* and *chisocs* as well. Suppose:

– $\Gamma = \{m \overset{?}{=}_{\mathsf{S} \cup \mathsf{A}} t\}$, is a $(\mathsf{S} \cup \mathsf{A})$-$\mathsf{UP}$ and suppose $m = m'\sigma_{comb}$, $t = t'\sigma_{comb}$, where $m' \in \mathit{SubTerms}(S_1)$;
– Variables in σ_{comb} are substituted with terms from the same semi-bundle:

$$(\forall x/X \in \sigma_{comb})((\exists i \in \{1, 2\})(x, X \in \mathit{SubTerms}(S_i))). \quad (10)$$

– Γ is $(\mathsf{S} \cup \mathsf{A})$-Unifiable.

Let $\tau \in U_{\mathsf{S} \cup \mathsf{A}}(\Gamma)$ and let A_{Th} denote a *Th*-UA. Using Def. 7 (**Combined Unifier**), say we have that $\tau \in \tau_{\mathsf{STD}} \odot \tau_{\mathsf{ACUN}}$ where $\tau_{\mathsf{STD}} \in A_{\mathsf{STD}}(\Gamma_{5.1})$ and $\tau_{\mathsf{ACUN}} \in A_{\mathsf{ACUN}}(\Gamma_{5.2})$.

Now from BSCA, if $m_1 \overset{?}{=}_{\mathsf{STD}} t_1 \in \Gamma_{5.1}$, and $\rho \in U_{\mathsf{STD}}(m_1 \overset{?}{=}_{\mathsf{STD}} t_1)$, then we have the following cases:

Variables. If m_1, and/or t_1 are variables, from (7) and BSCA, they are necessarily new i.e., $m_1, t_1 \in \mathit{Vars} \setminus \mathit{Vars}(\Gamma)$ (unless m and t are variables, which they are not, since *chcombcs* is normal). Hence, there are no new substitutions in ρ to $\mathit{Vars}(\Gamma)$ in this case.

Constants. If $m_1 \in \mathit{Constants}(S_1)$, again from BSCA, t_1 cannot belong to *Vars*, and it must be a constant. If m_1 is a fresh constant of S_1, then t_1 must also belong to S_1 from the freshness assumption (5) and (10), and if m_1 is not fresh, t_1 could belong to either $\mathit{SubTerms}(S_1)$ or *IIK* from Assumption 4. Further, $\rho = \{\}$.

Public Keys. If $m_1 = pk(\_)$, then t_1 must be some $pk(\_)$ as well. From BSCA, m_1 cannot be such that $[\_]^{\rightarrow}_{m_1} \sqsubset m$. Further, there cannot be an XOR term, say $\ldots \oplus m_1 \oplus \ldots$ that is a subterm of m, from μ-NUT Condition 2. The only other possibility is that $m = m_1$. In that case, t must also equal t_1, whence, t can belong to *IIK* from assumption 3 (Intruder possesses all public-keys). Hence, $(\forall x/X \in \rho)((\exists i \in \{1,2\})(x, X \in \mathit{SubTerms}(S_i)))$.

Encrypted Subterms. Suppose $m_1 = m_{11}\sigma_{comb}, t_1 = t_{11}\sigma_{comb}, m_{11}, t_{11} \in \mathit{EncSubt}(S_1 \cup S_2)$. Then, from μ-NUT Condition 1 and (6), we have, $m_{11}, t_{11} \in \mathit{EncSubt}(S_i)$, where $i \in \{1,2\}$. Hence, $(\forall x/X \in \rho)((\exists i \in \{1,2\})(x, X \in \mathit{SubTerms}(S_i)))$.

Sequences. If m_1 is a sequence, either m must be a sequence, or there must be some $\ldots \oplus m_1 \oplus \ldots$ belonging to $\mathit{SubTerms}(\{m,t\})$, from BSCA. But m and t cannot be sequences, since *chcombcs* is normal. Hence, by μ-NUT Condition 2 and (6), $m_1, t_1 \in \mathit{SubTerms}(S_i)\sigma_{comb}$, $i \in \{1,2\}$ and $(\forall x/X \in \rho)((\exists i \in \{1,2\})(x, X \in \mathit{SubTerms}(S_i)))$.

In summary, we make the following observations about problems in $\Gamma_{5.1}$.

If m_1 is an instantiation of a subterm in S_1, then so is t_1, or t_1 belongs to *IIK*:

$$(\forall m_1 \overset{?}{=}_{\mathsf{STD}} t_1 \in \Gamma_{5.1})(m_1 \in \mathit{SubTerms}(S_1)\sigma_{comb} \Rightarrow t_1 \in \mathit{SubTerms}(S_1)\sigma_{comb} \cup \mathit{IIK}). \tag{11}$$

Every substitution in τ_{STD} has both its term and variable from the same semi-bundle:

$$(\forall x/X \in \tau_{\mathsf{STD}})((\exists i \in \{1,2\})(x, X \in \mathit{SubTerms}(S_i))). \tag{12}$$

Now consider the UPs in $\Gamma_{5.2}$. Applying (7) into Lemma 1, we have that $\tau_{\mathsf{ACUN}} = \{\}$. Combining this with (12), we have:

$$(\forall x/X \in \tau)((\exists i \in \{1,2\})(x, X \in \mathit{SubTerms}(S_i)\sigma_{comb})). \tag{13}$$

Suppose $m = m_1 \oplus \ldots \oplus m_p$ and $t = t_1 \oplus \ldots \oplus t_q$; $p, q \geq 1$, $x = m\tau$, $y = t\tau$ and $m'' =_{\mathsf{S \cup A}} x$ where $m'' = m'_1 \oplus \ldots \oplus m'_{p'}$, s.t. $(\forall i, j \in \{1, \ldots, p'\})(i \neq j \Rightarrow m'_i\tau \neq_{\mathsf{S \cup A}} m'_j\tau)$ and $t'' =_{\mathsf{S \cup A}} y$, where $t'' = t'_1 \oplus \ldots \oplus t'_{q'}$, s.t. $(\forall i, j \in$

$\{1,\ldots,q'\})(i \neq j \Rightarrow t'_i\tau \neq_{\mathsf{S\cup A}} t'_j\tau)$. Informally, this means that, no two terms in $\{m'_1,\ldots,m'_{p'}\}$ or $\{t'_1,\ldots,t'_{q'}\}$ can be cancelled.

Suppose $\mathit{\Gamma}\psi = \mathit{\Gamma}_{5.1}$, where ψ is a set of substitutions. Then, $m\tau =_{\mathsf{S\cup A}} t\tau$ implies, $(\forall i \in \{1,\ldots,p'\})((\exists j \in \{1,\ldots,q'\})(m'_i\tau\psi =_{\mathsf{STD}} t'_j\tau\psi))$ with $p' = q'$. From (11), this means that $m \in \mathit{SubTerms}(S_1)\sigma_{comb}$ implies, t also belongs to $\mathit{SubTerms}(S_1)\sigma_{comb}$ or IIK.

Now since $\mathit{Vars}(m') \cup \mathit{Vars}(t') \subset \mathit{Vars}(S_1)$, we have, $m'\sigma_{comb} = m'\sigma_{iso}$, and $t'\sigma_{comb} = t'\sigma_{iso}$, where $\sigma_{comb} = \sigma_{iso} \cup \{x/X \mid x, X \in \mathit{SubTerms}(S_2)\}$. Combining this with (13), we have that, $m'\sigma_{comb}\tau =_{\mathsf{S\cup A}} t'\sigma_{comb}\tau \Rightarrow m'\sigma_{iso}\tau =_{\mathsf{S\cup A}} t'\sigma_{iso}\tau$.

Combining these with (2) and (3), we can now write:

$$(\forall \mathit{chcombcs}, \mathit{chisocs})\begin{pmatrix}\mathsf{childseq}(\mathit{chcombcs}, \mathit{combcs}, \mathsf{S}\cup\mathsf{A})\wedge\\ \mathsf{childseq}(\mathit{chisocs}, \mathit{isocs}, \mathsf{S}\cup\mathsf{A})\wedge\\ \mathsf{appl}(\mathsf{un}, \mathit{chcombcs}, \mathit{chcombcs}', \sigma_{comb}, \sigma'_{comb}, \mathsf{S}\cup\mathsf{A}) \Rightarrow\\ \mathsf{appl}(\mathsf{un}, \mathit{chisocs}, \mathit{chisocs}', \sigma_{iso}, \sigma'_{iso}, \mathsf{S}\cup\mathsf{A})\end{pmatrix}. \tag{14}$$

where, the active constraint in *chcombcs* and *chisocs* only differ in the term sets:

$$\begin{pmatrix}\mathsf{act}(m : \_ \cup t, \mathit{combcs}) \wedge \mathsf{act}(m : \_ \cup t, \mathit{isocs})\wedge\\ (\mathit{combcs}' = \mathit{combcs}_{<}\tau ^\frown \mathit{combcs}_{>}\tau) \wedge (\mathit{isocs}' = \mathit{isocs}_{<}\tau ^\frown \mathit{isocs}_{>}\tau)\wedge\\ (\sigma'_{comb} = \sigma_{comb} \cup \tau) \wedge (\sigma'_{iso} = \sigma_{iso} \cup \tau) \wedge (\tau \in (\mathsf{S}\cup\mathsf{A})\text{-}\mathit{mgu}(\langle m,\ t\rangle))\end{pmatrix}$$

Finally, we can combine, (5), (8), (9), and (14) to infer:

$$\begin{pmatrix}(\mathit{isocs} = \langle \_ : \_, \ldots, \_ : T\rangle) \wedge \mathsf{appl}(r_1, \mathit{isocs}^\frown[\mathit{sec} : T], \mathit{isocs}_1, \{\}, \sigma_1, \mathsf{S}\cup\mathsf{A})\wedge\\ \mathsf{appl}(r_2, \mathit{isocs}'_1, \mathit{isocs}_2, \sigma_1, \sigma_2, \mathsf{S}\cup\mathsf{A}) \wedge \ldots \wedge\\ \mathsf{appl}(r_p, \mathit{isocs}'_{p-1}, \mathit{isocs}_p, \sigma_{p-1}, \sigma_p, \mathsf{S}\cup\mathsf{A})\wedge\\ \mathsf{simple}(\mathit{isocs}_p) \wedge (\forall i \in \{1,\ldots,p\})(\mathit{isocs}'_i = \mathit{normalize}(\mathit{isocs}_i))\end{pmatrix}. \tag{15}$$

where $[r_1,\ldots,r_p]$ is a subsequence[3] of R (defined in 5).

This in turn implies $\mathsf{satisfiable}(\mathit{isocs}^\frown \mathit{sec} : T, \sigma_p, \mathsf{S}\cup\mathsf{A})$ from the definition of satisfiability.

We can then combine this with the fact that S_1 is a semi-bundle of P_1, and *isocs* is a constraint sequence of S_1 and conclude:

$$\mathsf{semi\text{-}bundle}(S_1, P_1) \wedge \mathsf{conseq}(\mathit{isocs}, S_1) \wedge (\mathit{isocs} = [\_ : \_, \ldots, \_ : T])\wedge\\ \mathsf{satisfiable}(\mathit{isocs}^\frown[\mathit{sec} : T], \sigma_p, \mathsf{S}\cup\mathsf{A}).$$

But from Definition 5 (**Secrecy**), this implies, $\neg\mathsf{secureForSecrecy}(P_1, \mathsf{S}\cup\mathsf{A})$, a contradiction to the hypothesis. Hence, P_1 is always secure for secrecy in the $(\mathsf{S}\cup\mathsf{A})$ theory, in combination with P_2 with which it is μ-NUT-*Satisfying*.

[3] s' is a *subsequence* of a sequence s, if $s = \_^\frown s'^\frown\_$.

5 Conclusion

In this paper, we provided a formal proof that tagging to ensure non-unifiability of distinct encryptions prevents multi-protocol attacks under the ACUN properties induced by the Exclusive-OR operator. We will now discuss some prospects for future work and related work.

5.1 Future work

Other equational theories can be handled in the same way as the ACUN theory: When we use BSCA, the UPs for them ($\Gamma_{5.2}$) will only have constants as subterms. Hence, unifiers only from the algorithms for standard theory problems need to be considered for μ-NUT-*Satisfying* protocols. Of course, this reasoning has to be given within a symbolic constraint solving model that takes the additional equational theories into account (the model we used, adapted from [11], was tailored to accommodate only ACUN).

We achieved our main result specifically for secrecy. The reason for this was that, in order to prove that attacks exist in isolation if there did in combination, we had to have a precise definition as to what an "attack" was to begin with. However, other properties such as authentication and observational equivalence can be considered on a case-by-case basis, with a similar proof pattern.

At the core of our proofs is the use of BSCA for combined theory unification. However, BSCA is applicable only for disjoint theories that do not share any operators. For instance, the algorithm cannot consider equations of the form, $[a, b] \oplus [c, d] = [a \oplus c, b \oplus d]$.

We plan to expand our proofs to include such equations in future, possibly with the help of new unification algorithms [12].

5.2 Related work

To the best of our knowledge, the consideration of algebraic properties and/or equational theories for protocol independence is unchartered waters.

A study of multi-protocol attacks with the perfect encryption assumption relaxed was first reported by Malladi et al. in [13] through "multi-protocol guessing attacks" on password protocols. Delaune et al. proved that these can be prevented by tagging in [14].

The original work of Guttman et al. in [1] assumed that protocols have no type-flaw attacks when they proved that tagging to ensure disjoint encryption prevents multi-protocol attacks. But a recent work by Guttman seems to relax that assumption [15]. Both [1] and [15] use the strand space model [7]. Our protocol model in this paper is also based on strand spaces, but the penetrator actions are modeled as symbolic reduction rules in the constraint solving algorithm of [11,10], as opposed to penetrator strands in [7]. Cortier-Delaune also prove that multi-protocol attacks can be prevented with tagging, which is slightly different from [1] and considers composed/non-atomic keys [16]. They too seem to use the constraints model as their protocol framework.

In [17], we prove the decidability of tagged protocols that use XOR with the underlying framework of [11] which extends [10] with XOR. That work is similar to our proofs, since we too used the same framework ([11]). Further, we use BSCA [6] as a core aspect of this paper along the lines of [17]. Recently, we used a similar proof pattern to prove that tagging prevents type-flaw attacks under XOR and most likely under other equational theories in [18]. Lemma 1 in the current paper was also the lynchpin in [18].

In [19], Kuesters and Truderung showed that the verification of protocols that use the XOR operator can be reduced to verification in a free term algebra, for a special class of protocols called $\oplus$-linear protocols[4], so that ProVerif can be used for verification. Chen et al. recently report some extensions to Kuesters-Truderung's work [20].

These results have a similarity with ours, in the sense that we too show that the algebraic properties of XOR have no effect when some of the messages are modified. However, we believe that our result is more general than these, since any protocol can be tagged to satisfy our requirements, but not necessarily $\oplus$-linearity.

Acknowledgments. I am thankful to Yannick Chevalier for explaining his protocol model in [11], Pascal Lafourcade for many useful remarks and the anonymous reviewers for their helpful suggestions.

References

1. Guttman, J.D., Thayer, F.J.: Protocol Independence through Disjoint Encryption. 13th IEEE Computer Security Foundations Workshop (2000) 24–34
2. Kelsey, J., Schneier, B., Wagner, D.: Protocol Interactions and the Chosen Protocol Attack. In: Proc. Security Protocols - 5th International Workshop, LNCS 1361 (1997) 91–104
3. Cremers, C.: Feasibility of multi-protocol attacks. In: First international conference on availability, reliability and security (ARES 2006), IEEE (2006) 287–294
4. Ryan, P.Y.A., Schneider, S.A.: An attack on a recursive authentication protocol. a cautionary tale. Inf. Process. Lett. **65**(1) (1998) 7–10
5. Tuengerthal, M.: Implementing a Unification Algorithm for Protocol Analysis with XOR. Technical Report 0609, Institut für Informatik, CAU Kiel, Germany (2006)
6. Baader, F., Schulz, K.U.: Unification in the union of disjoint equational theories: Combining decision procedures. J. of Symbolic Computation **21** (1996) 211–243
7. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: Why is a security protocol correct? In: Proc. IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1998) 160–171
8. Chevalier, Y., Küsters, R., Rusinowitch, M., Turuani, M.: An NP decision procedure for protocol insecurity with XOR. In: Proc. 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03), IEEE Computer Society Press (2003) 261–270

[4] Kuesters-Truderung define a term to be $\oplus$-linear if for each of its subterms of the form $s \oplus t$, either t or s is ground.

9. Cortier, V., Delaune, S.: Safely composing security protocols. Formal Methods in System Design (2008) To appear.
10. Millen, J., Shmatikov, V.: Constraint solving for bounded-process cryptographic protocol analysis. In: Proc. ACM Conference on Computer and Communication Security, ACM press (2001) 166–175
11. Chevalier, Y.: A simple constraint solving procedure for protocols with exclusive-or. Presented at Unif 2004 workshop (2004) available at http://www.lsv.ens-cachan.fr/unif/past/unif04/program.html.
12. Anantharaman, S., Lin, H., Lynch, C., Narendran, P., Rusinowitch, M.: Unification modulo homomorphic encryption. In: FroCos. (2009) 100–116
13. Malladi, S., Alves-Foss, J., Malladi, S.: What are multi-protocol guessing attacks and how to prevent them. In: 11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2002), IEEE Computer Society (2000) 77–82
14. Delaune, S., Kremer, S., Ryan, M.D.: Composition of password-based protocols. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08), Pittsburgh, PA, USA, IEEE Computer Society Press (2008) 239–251
15. Guttman, J.D.: Cryptographic protocol composition via the authentication tests. In: Foundations of Software Science and Computation Structures (FOSSACS, 2009), LNCS (2009)
16. Cortier, V., Delaune, S.: Safely composing security protocols. Formal Methods in System Design **34**(1) (2009) 1–36
17. Chevalier, Y., Malladi, S.: Decidability of "real-world" context-explicit security protocols. Tech. Report, DSU-SEED-CM07 (2007)
18. Malladi, S., Lafourcade, P.: How to prevent type-flaw attacks under algebraic properties. In: Security and Rewriting Techniques, Affiliated to CSF09 (2009)
19. Küsters, R., Truderung, T.: Reducing protocol analysis with xor to the xor-free case in the horn theory based approach. In: ACM Conference on Computer and Communications Security. (2008) 129–138
20. Chen, X., Deursen, T.V., Pang, J.: Improving automatic verification of security protocols with xor. In: 11th Conference on Formal Engineering Methods - ICFEM'09. (2009)

A Bader & Schulz Combined Theory Unification Algorithm (BSCA)

We will now consider how two UAs for two disjoint theories Th_1 and Th_2 respectively, may be combined to output the unifiers for UPs made using operators from $Th_1 \cup Th_2$ using Baader & Schulz Combination Algorithm (BSCA) [6].

We first need some definitions. Suppose F is a signature for a set of identities E and let Th denote the theory $=_E$. Then, a term is pure wrt Th iff every subterm of it is an F-term. i.e.,

$$\mathsf{pure}(t, Th) \Leftrightarrow (\forall t' \sqsubset t)((\exists f \in F)(t' = f(\_, \ldots, \_))).$$

We define a predicate ast (alien subterm) on terms such that, a term t' is an alien subterm of another term t wrt the theory Th, if it is a subterm of t, but is not pure wrt Th:

$$(\forall t, t', Th)(\mathsf{ast}(t', t, Th) \Leftrightarrow (t' \sqsubset t) \wedge \neg\mathsf{pure}(t', Th)).$$

For instance, $[1, n_a \oplus B, A]^{\rightarrow}_{pk(B)}$ has $n_a \oplus B$ as an alien subterm with respect to the theory STD.

We will use the following ($\mathsf{STD} \cup \mathsf{ACUN}$)-UP as our running example[5]:

$$\left\{[1, n_a]_{pk(B)} \overset{?}{=}_{\mathsf{STD}\cup\mathsf{ACUN}} [1, N_B]_{pk(a)} \oplus [2, A] \oplus [2, b]\right\}.$$

BSCA takes as input a ($Th_1 \cup Th_2$)-UP, say Γ, and applies some transformations on them to derive $\Gamma_{5.1}$ and $\Gamma_{5.2}$ that are Th_1-UP and Th_2-UP respectively.

Step 1 (Purify terms) BSCA first "purifies" the given set of ($Th = Th_1 \cup Th_2$)-UP, Γ, into a new set of problems Γ_1, such that, all the terms are pure wrt Th_1 or Th_2.

If our running example was Γ, then, the set of problems in Γ_1 are $W \overset{?}{=}_{\mathsf{STD}} [1, n_a]_{pk(B)}$, $X \overset{?}{=}_{\mathsf{STD}} [1, N_B]_{pk(a)}, Y \overset{?}{=}_{\mathsf{STD}} [2, A]$, $Z \overset{?}{=}_{\mathsf{STD}} [2, b]$, and $W \overset{?}{=}_{\mathsf{ACUN}} X \oplus Y \oplus Z$, where W, X, Y, Z are obviously new variables that did not exist in Γ.

Step 2. (Purify problems) Next, BSCA purifies Γ_1 into Γ_2 such that, every problem in Γ_2 has both terms pure wrt the same theory.

For our example problem, this step can be skipped since all the problems in Γ_1 already have both their terms purely from the same theory (STD or ACUN)).

[5] We omit the superscript $\rightarrow$ on encrypted terms in this problem, since they obviously use only asymmetric encryption.

Step 3. (Variable identification) Next, BSCA partitions $\mathit{Vars}(\mathit{\Gamma}_2)$ into a partition *VarIdP* such that, each variable in $\mathit{\Gamma}_2$ is replaced with a representative from the same equivalence class in *VarIdP*. The result is $\mathit{\Gamma}_3$.

In our example problem, one set of values for *VarIdP* can be

$$\{\{A\},\{B\},\{N_B\},\{W\},\{X\},\{Y,Z\}\}.$$

Step 4. (Split the problem) The next step of BSCA is to split $\mathit{\Gamma}_3$ into two UPs $\mathit{\Gamma}_{4.1}$ and $\mathit{\Gamma}_{4.2}$ such that, each of them has every problem with terms from the same theory, Th_1 or Th_2.

Following this in our example,

$$\mathit{\Gamma}_{4.1} = \left\{ W \overset{?}{=}_{\mathsf{STD}} [1, n_a]_{pk(B)}, X \overset{?}{=}_{\mathsf{STD}} [1, N_B]_{pk(a)}, Y \overset{?}{=}_{\mathsf{STD}} [2, A], Z \overset{?}{=}_{\mathsf{STD}} [2, b] \right\},$$

and

$$\mathit{\Gamma}_{4.2} = \left\{ W \overset{?}{=}_{\mathsf{ACUN}} X \oplus Y \oplus Y \right\}.$$

Step 5. (Solve systems) The penultimate step of BSCA is to partition all the variables in $\mathit{\Gamma}_3$ into a size of two: Let $p = \{V_1, V_2\}$ is a partition of $\mathit{Vars}(\mathit{\Gamma}_3)$. Then, the earlier problems $(\mathit{\Gamma}_{4.1}, \mathit{\Gamma}_{4.2})$ are further split such that, all the variables in one set of the partition are replaced with new constants in the other set and vice-versa. The resulting sets are $\mathit{\Gamma}_{5.1}$ and $\mathit{\Gamma}_{5.2}$.

In our sample problem, we can form $\{V_1, V_2\}$ as $\{\mathit{Vars}(\mathit{\Gamma}_3), \{\}\}$. i.e., we choose that all the variables in problems of $\mathit{\Gamma}_{5.2}$ be replaced with new constants. This is required to find the unifier for the problem (this is the partition that will successfully find a unifier).

So $\mathit{\Gamma}_{5.1}$ stays the same as $\mathit{\Gamma}_{4.1}$, but $\mathit{\Gamma}_{5.2}$ is changed to

$$\mathit{\Gamma}_{5.2} = \mathit{\Gamma}_{4.2}\beta = \left\{ W \overset{?}{=}_{\mathsf{ACUN}} X \oplus Y \oplus Y \right\} \beta = \left\{ w \overset{?}{=}_{\mathsf{ACUN}} x \oplus y \oplus y \right\}.$$

i.e., $\beta = \{w/W, x/X, y/Y\}$, where, w, x, y are constants, which obviously did not appear in $\mathit{\Gamma}_{5.1}$.

Step 6. (Combine unifiers) The final step of BSCA is to combine the unifiers for $\mathit{\Gamma}_{5.1}$ and $\mathit{\Gamma}_{5.2}$, obtained using A_{Th_1} and A_{Th_2}:

Definition 7. [Combined Unifier]

Let $\mathit{\Gamma}$ be a Th-UP where $(\mathit{Th}_1 \cup \mathit{Th}_2) = \mathit{Th}$. Let $\sigma_i \in A_{\mathit{Th}_i}(\mathit{\Gamma}_{5.i})$, $i \in \{1,2\}$ and let $V_i = \mathit{Vars}(\mathit{\Gamma}_{5.i})$, $i \in \{1,2\}$.

Suppose '$<$' is a linear order on $\mathit{Vars}(\mathit{\Gamma})$ such that $Y < X$ if X is not a subterm of an instantiation of Y:

$$(\forall X, Y \in \mathit{Vars}(\mathit{\Gamma}))((Y < X) \Rightarrow (\nexists\sigma)(X \sqsubset Y\sigma)).$$

Let $\mathsf{least}(X, T, <)$ *be defined as the minimal element of set* T*, when ordered linearly by the relation '*$<$*'. i.e.,*

$$\mathsf{least}(X, T, <) \Leftrightarrow (\forall Y \in T)((Y \neq X) \Rightarrow (X < Y)).$$

Then, the combined UA for Γ*, namely* $A_{Th_1 \cup Th_2}$*, is defined such that,*

$$A_{Th_1 \cup Th_2}(\Gamma) = \{\sigma \mid (\exists \sigma_1, \sigma_2)((\sigma = \sigma_1 \odot \sigma_2) \wedge (\sigma_1 \in A_{Th_1}(\Gamma_{5.1})) \wedge (\sigma_2 \in A_{Th_2}(\Gamma_{5.2})))\}.$$

where, if $\sigma = \sigma_1 \odot \sigma_2$*, then,*

– *The substitution in* σ *for the least variable in* V_1 *and* V_2 *is from* σ_1 *and* σ_2 *respectively:*

  $(\forall i \in \{1, 2\})((X \in V_i) \wedge \mathsf{least}(X, \mathit{Vars}(\Gamma), <) \Rightarrow (X\sigma = X\sigma_i))$*; and*

– *For all other variables* X*, where each* Y *with* $Y < X$ *has a substitution already defined, define* $X\sigma = X\sigma_i\sigma$ $(i \in \{1, 2\})$*:*

  $(\forall i \in \{1, 2\})((\forall X \in V_i)((\forall Y)((Y < X) \wedge (\exists Z)(Z/Y \in \sigma))) \Rightarrow (X\sigma = X\sigma_i\sigma))$.

B Proofs

The following lemma concerns combined unification problems involving STD and ACUN theories. We prove that, if we follow Bader & Schulz approach for finding unifiers for these problems, ACUN subproblems will have only constants as subterms. Consequently, we will end up in an empty set of substitutions returned by the ACUN UA for the ACUN UPs, even when the `XOR` terms are equal in the ACUN theory.

Lemma 2. [ACUN **UPs have only constants**]

Let $\Gamma = \{m \overset{?}{=}_{\mathsf{S} \cup \mathsf{A}} t\}$ *be a* $(\mathsf{S} \cup \mathsf{A})$*-UP that is* $(\mathsf{S} \cup \mathsf{A})$*-Unifiable, and where no subterm of* m *or* t *is an* ***XOR*** *term with free variables*[6]*:*

$$(\forall x)\left(\begin{matrix}((x \sqsubset m) \vee (x \sqsubset t)) \wedge (n \in \mathbb{N}) \wedge \\ (x = x_1 \oplus \ldots \oplus x_n)\end{matrix} \Rightarrow (\forall i \in \{1, \ldots, n\})(x_i \notin \mathit{Vars})\right).$$

Then,

$$(\forall m' \overset{?}{=}_{\mathsf{ACUN}} t' \in \Gamma_{5.2}; y)\left(\left(\begin{matrix}((y \sqsubset m') \vee (y \sqsubset t')) \wedge \\ (m' =_{\mathsf{ACUN}} t')\end{matrix}\right) \Rightarrow (y \in \mathit{Constants})\right).$$

[6] $\mathbb{N}$ is the set of natural numbers.

Proof. Let σ be a set of substitutions s.t. $\sigma \in A_{(\mathsf{S} \cup \mathsf{A})}(\Gamma)$.

Then, from Def. 7 (**Combined Unifier**), $\sigma \in \sigma_1 \odot \sigma_2$, where $\sigma_1 \in A_{\mathsf{STD}}(\Gamma_{5.1})$ and $\sigma_2 \in A_{\mathsf{ACUN}}(\Gamma_{5.2})$.

Suppose there is a term t in Γ with an alien subterm t' wrt the theory ACUN (e.g. $[1, n_a]_k^{\rightarrow} \oplus b \oplus c$ with the alien subterm of $[1, n_a]_k^{\rightarrow}$).

Then, from the definition of Γ_2, it must have been replaced with a new variable in Γ_2. i.e.,

$$(\forall t, t') \left(\begin{pmatrix} (t \in \Gamma) \wedge (t = \_ \oplus \ldots \oplus \_) \wedge \\ (t' \sqsubset t) \wedge \mathsf{ast}(t', t, \mathsf{ACUN}) \end{pmatrix} \Rightarrow (\exists X) \begin{pmatrix} (X \overset{?}{=}_{\mathsf{ACUN}} t' \in \Gamma_2) \wedge \\ (X \in \mathit{NewVars}) \end{pmatrix} \right). \tag{16}$$

where $\mathit{NewVars} \subset \mathit{Vars} \setminus \mathit{Vars}(\Gamma)$.

Since `XOR` terms do not have free variables from hypothesis, it implies that every free variable in an `XOR` term in Γ_2 is a new variable:

$$(\forall t, t') \left(\begin{pmatrix} (t \in \Gamma_2) \wedge \mathsf{pure}(t, \mathsf{ACUN}) \wedge \\ (t' \sqsubset t) \wedge (t' \in \mathit{Vars}) \end{pmatrix} \Rightarrow (t' \in \mathit{NewVars}) \right). \tag{17}$$

Since every alien subterm of every term in Γ has been replaced with a new variable (16), combining it with (17), `XOR` terms in Γ_2 must now have only constants and/or new variables:

$$(\forall t, t') \left(\begin{pmatrix} \mathsf{pure}(t, \mathsf{ACUN}) \wedge \\ (t \in \Gamma_2) \wedge (t' \sqsubset t) \end{pmatrix} \Rightarrow (t' \in \mathit{NewVars} \cup \mathit{Constants}) \right). \tag{18}$$

Let VarIdP be a partition of $\mathit{Vars}(\Gamma_2)$ and $\Gamma_3 = \Gamma_2\rho$, such that

$$\Gamma_2\rho = \{ s \overset{?}{=}_{\mathsf{ACUN}} t \mid (s \overset{?}{=}_{\mathsf{ACUN}} t := s'\rho \overset{?}{=}_{\mathsf{ACUN}} t'\rho) \wedge s' \overset{?}{=}_{\mathsf{ACUN}} t' \in \Gamma \}$$

where ρ is the set of substitutions where each set of variables in VarIdP has been replaced with one of the variables in the set:

$$\rho = \left\{ x/X \mid \left((\forall Y_1/X_1, Y_2/X_2 \in \rho; \mathit{vip} \in \mathit{VarIdP}) \begin{pmatrix} (X_1, X_2 \in \mathit{vip}) \Rightarrow \\ (Y_1 = Y_2) \wedge \\ (Y_1, Y_2 \in \mathit{vip}) \end{pmatrix} \right) \right\}.$$

Can there exist a substitution X/Y in ρ such that $Y \in \mathit{NewVars}$ and $X \in \mathit{Vars}(\Gamma)$?

To find out, consider the following two statements:

– From (16), every new variable Y in Γ_2 belongs to a STD-UP in Γ_2:

$$(\forall Y \in \mathit{NewVars})((Y \in \mathit{Vars}(\Gamma_2) \Rightarrow (\exists t)(\mathsf{pure}(t, \mathsf{STD}) \wedge Y \overset{?}{=}_{\mathsf{ACUN}} t \in \Gamma_2))).$$

– Further, from hypothesis, we have that $\mathtt{XOR}$ terms in Γ do not have free variables. Hence, every free variable is a proper subterm[7] of a purely STD term:

$$(\forall X \in \mathit{Vars}(\Gamma))\left((\exists t \in \Gamma)((X \sqsubset t) \wedge \mathsf{pure}(t, \mathsf{STD}) \wedge (X \neq t))\right).$$

The above two statements are contradictory: It is not possible that a new variable and an existing variable can be replaced with each other, since one belongs to a STD-UP, and another is always a proper subterm of a term that belongs to a STD-UP.

Hence, *VarIdP* cannot consist of sets where new variables are replaced by $\mathit{Vars}(\Gamma)$. i.e.,

$$(\nexists X, Y; \mathit{vip} \in \mathit{VarIdP})\begin{pmatrix}(Y, X \in \mathit{vip}) \wedge (Y \in \mathit{NewVars}) \wedge \\ (X \in \mathit{Vars}(\Gamma)) \wedge (X/Y \in \rho)\end{pmatrix} \tag{19}$$

Writing (19) in (18), we have,

$$(\forall t, t')\left(\begin{pmatrix}\mathsf{pure}(t, \mathsf{ACUN}) \wedge \\ (t \in \Gamma_3) \wedge (t' \sqsubset t)\end{pmatrix} \Rightarrow (t' \in \mathit{NewVars} \cup \mathit{Constants})\right). \tag{20}$$

Further, if a variable belongs to a UP of Γ_3, then the other term of the UP is pure wrt STD theory:

$$(\forall X \in \mathit{Vars}(\Gamma_3), t)\left(\begin{pmatrix}(X \overset{?}{=}_{\mathsf{ACUN}} t \in \Gamma_3) \vee \\ (t \overset{?}{=}_{\mathsf{ACUN}} X \in \Gamma_3)\end{pmatrix} \Rightarrow (X \in \mathit{NewVars}) \wedge \mathsf{pure}(t, \mathsf{STD})\right). \tag{21}$$

Now suppose $\Gamma_{4.2} = \{s \overset{?}{=}_{\mathsf{ACUN}} t \mid (s \overset{?}{=}_{\mathsf{ACUN}} t \in \Gamma_3) \wedge \mathsf{pure}(s, \mathsf{ACUN}) \wedge \mathsf{pure}(t, \mathsf{ACUN})\}$, $\{V_1, V_2\}$ a partition of $\mathit{Vars}(\Gamma) \cup \mathit{NewVars}$, and

$$\Gamma_{5.2} = \Gamma_{4.2}\beta,$$

where, β is a set of substitutions of new constants to V_1:

$$\beta = \{x/X \mid (X \in V_1) \wedge (x \in \mathit{Constants} \setminus (\mathit{Constants}(\Gamma) \cup \mathit{Constants}(\Gamma_{5.1})))\}.$$

From hypothesis, $\Gamma_{5.2}$ is ACUN-Unifiable. Hence, we have:

$$(\forall \sigma)((\forall m' \overset{?}{=}_{\mathsf{ACUN}} t' \in \Gamma_{5.2})(m'\sigma =_{\mathsf{ACUN}} t'\sigma) \Leftrightarrow \sigma \in A_{\mathsf{ACUN}}(\Gamma_{5.2})).$$

Now consider a σ s.t. $\sigma \in A_{\mathsf{ACUN}}(\Gamma_{5.2})$.

From (20), we have that $\mathtt{XOR}$ terms in $\Gamma_{5.2}$ have only new variables and/or constants and from (21) we have that if $X \in \mathit{Vars}(\Gamma_{5.2})$, then there exists t s.t. $X \overset{?}{=}_{\mathsf{STD}} t \in \Gamma_{5.1}$ and t is pure wrt STD theory.

[7] t is a proper subterm of t' if $t \sqsubset t' \wedge t \neq t'$.

Suppose $V_2 \neq \{\}$. Then, there is at least one variable, say $X \in \mathit{Vars}(\mathit{\Gamma}_{5.2})$. This implies that X is replaced with a constant (say x) in $\mathit{\Gamma}_{5.1}$.

Since X is necessarily a new variable and one term of a STD-UP, this implies that x must equal some compound term made with *StdOps*.

However, a compound term made with *StdOps* can never equal a constant under the STD theory:

$$(\nexists f \in \mathit{StdOps}; t_1, \ldots, t_n; x \in \mathit{Constants})(x =_{\mathsf{STD}} f(t_1, \ldots, t_n)),$$

a contradiction.

Hence, $\sigma = \{\}$, $V_2 = \{\}$ and our hypothesis is true that all $\mathtt{XOR}$ terms in $\mathit{\Gamma}_{5.2}$ necessarily contain only constants:

$$(\forall m' \overset{?}{=}_{\mathsf{ACUN}} t' \in \mathit{\Gamma}_{5.2}; x) \left((x \sqsubset m) \vee (x \sqsubset t) \Rightarrow (x \in \mathit{Constants}) \right).$$